FAST ALTERNATING LINEARIZATION METHODS FOR MINIMIZING THE SUM OF TWO CONVEX FUNCTIONS

DONALD GOLDFARB[*], SHIQIAN MA[*], AND KATYA SCHEINBERG[†]

October 11, 2010

Abstract. We present in this paper first-order alternating linearization algorithms based on an alternating direction augmented Lagrangian approach for minimizing the sum of two convex functions. Our basic methods require at most $O(1/\epsilon)$ iterations to obtain an ϵ-optimal solution, while our accelerated (i.e., fast) versions of them require at most $O(1/\sqrt{\epsilon})$ iterations, with little change in the computational effort required at each iteration. For both types of methods, we present one algorithm that requires both functions to be smooth with Lipschitz continuous gradients and one algorithm that needs only one of the functions to be so. Algorithms in this paper are Gauss-Seidel type methods, in contrast to the ones proposed by Goldfarb and Ma in [21] where the algorithms are Jacobi type methods. Numerical results are reported to support our theoretical conclusions and demonstrate the practical potential of our algorithms.

1. Introduction. In this paper, we are interested in the following convex optimization problem:

$$\min F(x) \equiv f(x) + g(x) \tag{1.1}$$

where $f, g : \mathbb{R}^n \to \mathbb{R}$ are both convex functions such that the following two problems are easy to solve for any $\tau > 0$ and $z \in \mathbb{R}^n$ relative to minimizing $F(x)$:

$$\min \left\{ \tau f(x) + \frac{1}{2}\|x - z\|^2 \right\} \tag{1.2}$$

and

$$\min \left\{ \tau g(x) + \frac{1}{2}\|x - z\|^2 \right\}. \tag{1.3}$$

In particular, we are specially interested in cases where solving (1.2) (or (1.3)) takes roughly the same effort as computing the gradient (or a subgradient) of $f(x)$ (or $g(x)$, respectively). Problems of this type arise in many applications of practical interest. The following are some interesting examples.

Example 1. ℓ_1 minimization in compressed sensing (CS). Signal recovery problems in compressed sensing [8, 13] use the ℓ_1 norm $\|x\|_1 := \sum_{i=1}^n |x_i|$ as a regularization term to enforce sparsity in the solution $x \in \mathbb{R}^n$ of a linear system $Ax = b$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. This results in the unconstrained problem

$$\min \left\{ \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1 \right\}, \tag{1.4}$$

[*]Department of Industrial Engineering and Operations Research, Columbia University, New York, 10027, USA. Email: {goldfarb, sm2756}@columbia.edu. Research supported in part by NSF Grants DMS 06-06712 and DMS 10-16571, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER25856.

[†]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA. Email: katyas@lehigh.edu

where $\rho > 0$, which is of the form of (1.1) with $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) := \rho\|x\|_1$. In this case, the two problems (1.2) and (1.3) are easy to solve. Specifically, (1.2) reduces to solving a linear system and (1.3) reduces to a vector shrinkage operation which requires $O(n)$ operations (see e.g., [23]). Depending on the size and structure of A solving the system of linear equations required by (1.2) may be more expensive, less expensive or comparable to computing the gradient $A^\top(Ax - b)$ of $f(x)$. In the application we consider in Section 4 these computations are comparable due to the special structure of A.

Example 2. Nuclear norm minimization (NNM). The nuclear norm minimization problem, which seeks a low-rank solution of a linear system, can be cast as

$$(1.5) \qquad \min\left\{\frac{1}{2}\|\mathcal{A}(X) - b\|_2^2 + \rho\|X\|_*\right\},$$

where $\tau > 0$, $X \in \mathbb{R}^{m\times n}$, $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ is a linear operator, $b \in \mathbb{R}^p$ and the nuclear norm $\|X\|_*$ is defined as the sum of the singular values of the matrix X. Problem (1.5) and a special case of it, the so-called matrix completion problem, have many applications in optimal control, online recommendation systems, computer vision, etc. (see e.g., [7, 9, 27, 42]). In Problem (1.5), if we let $f(X) = \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2$ and $g(X) = \rho\|X\|_*$, then problem (1.2) reduces to solving a linear system. Problem (1.3) has a closed-form solution that is given by matrix shrinkage operation (see e.g., [34]).

Example 3. Robust principal component analysis (RPCA). The RPCA problem seeks to recover a low-rank matrix X from a corrupted matrix M. This problem has many applications in computer vision, image processing and web data ranking (see e.g., [6]), and can be formulated as

$$(1.6) \qquad \min\left\{\|X\|_* + \rho\|Y\|_1 : X + Y = M\right\},$$

where $\rho > 0$, $M \in \mathbb{R}^{m\times n}$ and the ℓ_1 norm $\|Y\|_1 := \sum_{i,j}|Y_{ij}|$. Note that (1.6) can be rewritten as

$$\min\{\|X\|_* + \rho\|M - X\|_1\},$$

which is of the form of (1.1). Moreover, the two problems (1.2) and (1.3) corresponding to (1.6) have closed-form solutions given respectively by a matrix shrinkage operation and a vector shrinkage operation. The matrix shrinkage operation requires a singular value decomposition (SVD) and is comparable in cost to computing a subgradient of $\|X\|_*$ or the gradient of the smoothed version of this function (see Section 5).

Example 4. Sparse inverse covariance selection (SICS). Gaussian graphical models are of great interest in statistical learning. Because conditional independence between different nodes correspond to zero entries in the inverse covariance matrix of the Gaussian distribution, one can learn the structure of the graph by estimating a sparse inverse covariance matrix from sample data by solving the following maximum likelihood problem with an ℓ_1-regularization term, (see e.g., [3, 18, 49, 52]).

$$\max\left\{\log\det(X) - \langle \Sigma, X \rangle - \rho\|X\|_1\right\},$$

or equivalently,

$$(1.7) \qquad \min\left\{-\log\det(X) + \langle \Sigma, X \rangle + \rho\|X\|_1\right\},$$

where $\rho > 0$ and $\Sigma \in S_+^n$ (the set of symmetric positive semidefinite matrices) is the sample covariance matrix. Note that by defining $f(X) := -\log\det(X) + \langle \Sigma, X \rangle$ and $g(X) := \rho\|X\|_1$, (1.7) is of the form of (1.1). Moreover, it can be proved that the problem (1.2) has a closed-form solution, which is given by a spectral decomposition - a comparable effort to computing the gradient of $f(X)$, while the solution of problem (1.3) corresponds to a vector shrinkage operation.

Algorithms for solving problem (1.1) have been studied extensively in the literature. For large-scale problems, for which problems (1.2) and (1.3) are relatively easy to solve, the class of alternating direction methods that are based on variable splitting combined with the augmented Lagrangian method are particularly important. In these methods, one splits the variable x into two variables, i.e., one introduces a new variable y and rewrites Problem (1.1) as

$$(1.8) \qquad \min\{f(x) + g(y) : x - y = 0\}.$$

Since Problem (1.8) is an equality constrained problem, the augmented Lagrangian method can be used to solve it. Given a penalty parameter $1/\mu$, at the k-th iteration, the augmented Lagrangian method minimizes the augmented Lagrangian function

$$(1.9) \qquad \mathcal{L}_\mu(x, y; \lambda) := f(x) + g(y) - \langle \lambda, x - y \rangle + \frac{1}{2\mu}\|x - y\|^2,$$

with respect to x and y, i.e., it solves the subproblem

$$(1.10) \qquad (x^k, y^k) := \arg\min_{x,y} \mathcal{L}_\mu(x, y; \lambda^k)$$

and then updates the Lagrange multiplier λ^k via:

$$(1.11) \qquad \lambda^{k+1} := \lambda^k - \frac{1}{\mu}(x^k - y^k).$$

Minimizing $\mathcal{L}_\mu(x, y; \lambda)$ with respect to x and y jointly is often not easy. In fact, it certainly is not any easier than solving the original problem (1.1). However, if one minimizes $\mathcal{L}_\mu(x, y; \lambda)$ with respect to x and y alternatingly, one needs to solve problems of the form (1.2) and (1.3), which as we have already discussed, is often easy to do. Such an alternating direction augmented Lagrangian method (ADAL) for solving (1.8) is given below as Algorithm 1.

Algorithm 1: Alternating Direction Augmented Lagrangian Method (ADAL)

---
1 Choose μ, λ^0 and $x^0 = y^0$.
2 for $k = 0, 1, \cdots$ do
3 $\quad$ $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$
4 $\quad$ $y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^k)$
5 $\quad$ $\lambda^{k+1} := \lambda^k - \frac{1}{\mu}(x^{k+1} - y^{k+1})$

The history of alternating direction methods (ADMs) goes back to the 1950s for solving PDEs [14, 41] and to the 1970s for solving variational problems associated with PDEs [19, 20]. ADMs have also been applied to solving variational inequality problems by Tseng [46, 47] and He et al. [24, 26]. Recently, with the emergence of compressive sensing and subsequent great interest in ℓ_1 minimization [8, 13], ADMs have been applied to

ℓ_1 and total variation regularized problems arising from signal processing and image processing. The papers of Goldstein and Osher [22], Afonso et al. [2] and Yang and Zhang [51] are based on the alternating direction augmented Lagrangian framework (Algorithm 1), and demonstrate that ADMs are very efficient for solving ℓ_1 and TV regularized problems. The work of Yuan [53] and Yuan and Yang [54] showed that ADMs can also efficiently solve ℓ_1-regularized problems arising from statistics and data analysis. More recently, Wen, Goldfarb and Yin [50] and Malick et al. [35] applied alternating direction augmented Lagrangian methods to solve semidefinite programming (SDP) problems. The results in [50] show that these methods greatly outperform interior point methods on several classes of well-structured SDP problems. Furthermore, He et al. proposed an alternating direction based contraction method for solving separable linearly constrained convex problems [25].

Another important and related class of algorithms for solving (1.1) is based on operator-splitting. The aim of these algorithms is to find an x such that

$$(1.12) \qquad 0 \in T_1(x) + T_2(x),$$

where T_1 and T_2 are maximal monotone operators. This is a more general problem than (1.1) and ADMs for it have been the focus of a substantial amount of research; e.g., see [10, 11, 12, 15, 16, 32, 44]. Since the first-order optimality conditions for (1.1) are:

$$(1.13) \qquad 0 \in \partial f(x) + \partial g(x),$$

where $\partial f(x)$ denotes the subdifferential of $f(x)$ at the point x, a solution to Problem (1.1) can be obtained by solving Problem (1.12). For example, see [15, 32, 44] and references therein for more information on this class of algorithms.

While global convergence results for various splitting and alternating direction algorithms have been established under appropriate conditions, our interest here is on iteration complexity bounds for such algorithms. By an iteration complexity bound we mean a bound on the number of iterations needed to obtain an ϵ-optimal solution which is defined as follows.

DEFINITION 1.1. $x_\epsilon \in \mathbb{R}^n$ is called an ϵ-optimal solution to (1.1) if $F(x_\epsilon) - F(x^*) \leq \epsilon$, where x^* is an optimal solution to (1.1).

Complexity bounds for first-order methods for solving convex optimization problems have been given by Nesterov and many others. In [37, 38], Nesterov gave first-order algorithms for solving smooth unconstrained convex minimization problems with an iteration complexity of $O(\sqrt{L/\epsilon})$, where L is the Lipschitz constant of the gradient of the objective function, and showed that this is the best complexity that is obtainable when only first-order information is used. These methods can be viewed as accelerated gradient methods where a combination of past iterates are used to compute the next iterate. Similar techniques were then applied to nonsmooth problems [4, 39, 40, 48] and corresponding optimal complexity results were obtained. The ISTA (Iterative Shrinkage/Thresholding Algorithm) and FISTA (Fast Iterative Shrinkage/Thresholding Algorithm) algorithms proposed by Beck and Teboulle in [4] are designed for solving (1.1) when one of the functions (say $f(x)$) is smooth and the other is not. It is proved in [4] that the number of iterations required by ISTA and FISTA to get an ϵ-optimal solution to problem (1.1) are respectively $O(L(f)/\epsilon)$ and $O(\sqrt{L(f)/\epsilon})$, under the assumption that $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant $L(f)$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L(f)\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n.$$

ISTA computes a sequence $\{x^k\}$ via the iteration

(1.14)
$$x^{k+1} := \arg\min_x Q_f(x, x^k),$$

where

(1.15)
$$Q_f(u, v) := g(u) + f(v) + \langle \nabla f(v), u - v \rangle + \frac{1}{2\mu} \|u - v\|^2,$$

while FISTA computes $\{x^k\}$ via the iteration

(1.16)
$$\begin{cases} x^k & := \arg\min_x Q_f(x, y^k) \\ t_{k+1} & := \left(1 + \sqrt{1 + 4t_k^2}\right)/2 \\ y^{k+1} & := x^k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x^k - x^{k-1}) \end{cases}$$

starting with $t_1 = 1$, $y^1 = x^0 \in \mathbb{R}^n$ and $k = 1$.

Note that ISTA and FISTA treat the functions $f(x)$ and $g(x)$ very differently. At each iteration they both linearize the function $f(x)$ but never directly minimize it, while they do minimize the function $g(x)$ in conjunction with the linearization of $f(x)$ and a proximal (penalty) term. These two methods have proved to be efficient for solving the CS problem (1.4) (see e.g., [4, 23]) and the NNM problem (1.5) (see e.g., [34, 45]). ISTA and FISTA work well in these areas because $f(x)$ is quadratic and is well approximated by linearization. However, for the RPCA problem (1.6) where two complicated functions are involved, ISTA and FISTA do not work well. As we shall show in Sections 2 and 3, our ADMs are very effective in solving RPCA problems. For the SICS problem (1.7), intermediate iterates X^k may not be positive definite, and hence the gradient of $f(X) = -\log\det(X) + \langle \Sigma, X \rangle$ may not be well defined at X^k. Therefore, ISTA and FISTA cannot be used to solve the SICS problem (1.7). In [43], it is shown that SICS problems can be very efficiently solved by our ADM approach.

Our contribution. In this paper, we propose both basic and accelerated (i.e., *fast*) versions of first-order alternating linearization methods (ALMs) based on an alternating direction augmented Lagrangian approach for solving (1.1) and analyze their iteration complexities. Our basic methods require at most $O(L/\epsilon)$ iterations to obtain an ϵ-optimal solution, while our fast methods require at most $O(\sqrt{L/\epsilon})$ iterations with only a very small increase in the computational effort required at each iteration. Thus, our fast methods are *optimal* first-order methods in terms of iteration complexity. For both types of methods, we present an algorithm that requires both functions to be continuously differentiable with Lipschitz constants for the gradients denoted by $L(f)$ and $L(g)$. In this case $L = \max\{L(f), L(g)\}$. We also present for each type of method, an algorithm that only needs one of the functions, say $f(x)$, to be smooth, in which case $L = L(f)$. These algorithms are related to the multiple splitting algorithms in a recent paper by Goldfarb and Ma [21]. The algorithms in [21] are Jacobi type methods since they do not use information from the current iteration to solve succeeding subproblems in that iteration, while the algorithms proposed in this paper are Gauss-Seidel type methods since information from the current iteration is used later in the same iteration. These algorithms can also be viewed as extensions of the ISTA and FISTA algorithms in [4]. The complexity bounds we obtain for our algorithms are similar to (and as much as a factor of two better that) those in [4].

At each iteration, our algorithms alternatively minimize two different approximations to the original objective function, obtained by keeping one function unchanged and linearizing the other one. Our basic algorithm is similar in many ways to the alternating linearization method proposed by Kiwiel et al. [28]. In

particular, the approximate functions minimized at each step of Algorithm 3.1 in [28] have the same form as those minimized in our algorithm. However, our basic algorithm differs from the one in [28] in the way that the proximal terms are chosen, and our accelerated algorithms are very different. Moreover, no complexity bounds have been given for the algorithm in [28]. To the best of our knowledge, the complexity results in this paper are the first ones that have been given for a Gauss-Seidel type alternating direction method [1]. Complexity results for related Jacobi type alternating direction methods are given in [21].

Organization. The rest of this paper is organized as follows. In Sections 2 and 3 we propose our alternating linearization methods based on alternating direction augmented Lagrangian methods and give convergence/complexity bounds for them. We compare the performance of our ALMs to other competing first-order algorithms using an image deblurring problem in Section 4. In Section 5, we apply our ALMs to solve very large RPCA problems arising from background extraction in surveillance video and matrix completion and report the numerical results. Finally, we make some conclusion in Section 6.

2. Alternating Linearization Methods. In iteration of the ADAL method, Algorithm 1, the Lagrange multiplier λ is updated just once, immediately after the augmented Lagrangian is minimized with respect to y. Since the alternating direction approach is meant to be symmetric with respect to x and y, it is natural to also update λ after solving the subproblem with respect to x. By doing this, we get a symmetric version of the ADAL method. This algorithm is given below as Algorithm 2.

Algorithm 2: Symmetric Alternating Direction Augmented Lagrangian Method (SADAL)

---

1 Choose μ, λ^0 and $x^0 = y^0$.

2 for $k = 0, 1, \cdots$ do

3 $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$

4 $\lambda^{k+\frac{1}{2}} := \lambda^k - \frac{1}{\mu}(x^{k+1} - y^k)$

5 $y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}})$

6 $\lambda^{k+1} := \lambda^{k+\frac{1}{2}} - \frac{1}{\mu}(x^{k+1} - y^{k+1})$

This ADAL variant is described and analyzed in [20]. Moreover, it is shown in [20] that Algorithms 1 and 2 are equivalent to the Douglas-Rachford [14] and Peaceman-Rachford [41] methods, respectively applied to the optimality condition (1.13) for problem (1.1). If we assume that both $f(x)$ and $g(x)$ are differentiable, it follows from the first order optimality conditions for the two subproblems in lines 3 and 5 of Algorithm 2 that

$$(2.1) \qquad \lambda^{k+\frac{1}{2}} = \nabla f(x^{k+1}) \quad \text{and} \quad \lambda^{k+1} = -\nabla g(y^{k+1}).$$

Substituting (2.1) into Algorithm 2, we get the following alternating linearization method (ALM) which is equivalent to the SADAL method (Algorithm 2) when both f and g are differentiable. In Algorithm 3, $Q_f(u, v)$ is defined by (1.15) and

$$(2.2) \qquad Q_g(u, v) := f(u) + g(v) + \langle \nabla g(v), u - v \rangle + \frac{1}{2\mu}\|u - v\|_2^2.$$

In Algorithm 3, we alternatively replace the functions g and f by their linearizations plus a proximal

---

[1] After completion of an earlier version of the present paper, which is available on http://arxiv.org/abs/0912.4571, Monteiro and Svaiter [36] gave an iteration complexity bound to achieve a desired closeness of the current iterate to the solution for ADMs for solving the more general problem (1.12).

Algorithm 3: Alternating Linearization Method (ALM)

---

1 Choose μ and $x^0 = y^0$.

2 **for** $k = 0, 1, \cdots$ **do**

3 $\quad$ $x^{k+1} := \arg\min_x Q_g(x, y^k)$

4 $\quad$ $y^{k+1} := \arg\min_y Q_f(y, x^{k+1})$

---

regularization term to get an approximation to the original function F. Thus, our ALM algorithm can also be viewed as a proximal point algorithm.

A drawback of Algorithm 3 is that it requires both f and g to be continuously differentiable. In many applications, however, one of these functions is nonsmooth, as in the examples given in Section 1. Although Algorithm 2 can be applied when $f(x)$ and $g(x)$ are nonsmooth, we are unable to provide a comparable complexity bound in this case. However, when only one of the functions of f and g is nonsmooth (say g is nonsmooth), the following variant of Algorithm 3 applies, and for this algorithm, we have a complexity result.

Algorithm 4: Alternating Linearization Method with Skipping Steps (ALM-S)

---

1 Choose μ, λ^0 and $x^0 = y^0$.

2 **for** $k = 0, 1, \cdots$ **do**

3 $\quad$ $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$

4 $\quad$ **If** $F(x^{k+1}) > \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$, **then** $x^{k+1} := y^k$

5 $\quad$ $y^{k+1} := \arg\min_y Q_f(y, x^{k+1})$

6 $\quad$ $\lambda^{k+1} := \nabla f(x^{k+1}) - (x^{k+1} - y^{k+1})/\mu$

We call Algorithm 4, ALM with skipping steps (ALM-S) because in line 4 of Algorithm 4, if

$$(2.3) \qquad\qquad\qquad F(x^{k+1}) > \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$$

holds, we let $x^{k+1} := y^k$, i.e., we skip the computation of x^{k+1} in line 3. An alternative version of Algorithm 4 that has smaller average work per iteration is the following Algorithm 5.

Algorithm 5: Alternating Linearization Method with Skipping Steps (equivalent version)

---

1 Choose μ, λ^0 and $x^0 = y^0$.

2 **for** $k = 0, 1, \cdots$ **do**

3 $\quad$ $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$

4 $\quad$ **if** $F(x^{k+1}) > \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$ **then**

5 $\quad\quad$ $x^{k+1} := y^k$

6 $\quad\quad$ $y^{k+1} := \arg\min_y Q_f(y, x^{k+1})$

7 $\quad\quad$ $\lambda^{k+1} := \nabla f(x^{k+1}) - (x^{k+1} - y^{k+1})/\mu$

8 $\quad$ **else**

9 $\quad\quad$ $\lambda^{k+\frac{1}{2}} := \lambda^k - (x^{k+1} - y^k)/\mu$

10 $\quad\quad$ $y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}})$

11 $\quad\quad$ $\lambda^{k+1} := \lambda^{k+\frac{1}{2}} - (x^{k+1} - y^{k+1})/\mu$

Note that in Algorithm 5, when (2.3) does not hold, we switch to the SADAL algorithm, which updates λ instead of computing $\nabla f(x^{k+1})$. Algorithm 5 is usually faster than Algorithm 4 when $\nabla f(x^{k+1})$ is costly

to compute in addition to performing Step 3. Note also that when (2.3) holds, then the steps of the algorithm reduce to those of ISTA.

The following theorem gives conditions under which Algorithms 2, 3, 4 and 5 are equivalent.

THEOREM 2.1. *(i) If both f and g are differentiable, and λ_0 is set to $-\nabla g(y^0)$, then Algorithms 2 and 3 are equivalent. (ii) If in addition g is Lipschitz continuous with Lipschitz constant $L(g)$, and $\mu \leq 1/L(g)$, then Algorithms 3 and 4 are equivalent. (iii) If f is differentiable, then Algorithms 4 and 5 are equivalent.*

Proof. When both f and g are differentiable and $\lambda_0 = -\nabla g(y^0)$, (2.1) holds for all $k \geq 0$, and it follows that $\mathcal{L}_\mu(x, y^k; \lambda^k) \equiv Q_g(x, y^k)$ and $\mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}}) \equiv Q_f(y, x^{k+1})$. This proves part (i). If $\nabla g(x)$ is Lipschitz continuous and $\mu \leq 1/L(g)$,

$$g(x^{k+1}) \leq g(y^k) + \langle \nabla g(y^k), x^{k+1} - y^k \rangle + \frac{1}{2\mu} \left\| x^{k+1} - y^k \right\|_2^2,$$

holds (see e.g., [5]). This implies that (2.3) does not hold and hence, $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$, and the equivalence of Algorithms 3 and 4 follows. This proves part (ii). The optimality of x^{k+1} in line 3 of Algorithm 5 implies that $\lambda^{k+\frac{1}{2}} = \nabla f(x^{k+1})$ when (2.3) does not hold and hence that $\mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}}) \equiv Q_f(y, x^{k+1})$. This proves part (iii). $\square$

We show in the following that the iteration complexity of Algorithm 4 is $O(1/\epsilon)$ for obtaining an ϵ-optimal solution for (1.1). First, we need the following generalization of Lemma 2.3 in [4].

LEMMA 2.2. *Let $\psi : \mathbb{R}^n \to \mathbb{R}$ and $\phi : \mathbb{R}^n \to \mathbb{R}$ be convex functions and define*

$$Q_\psi(u, v) := \phi(u) + \psi(v) + \langle \gamma_\psi(v), u - v \rangle + \frac{1}{2\mu} \|u - v\|_2^2,$$

and

$$(2.4) \qquad\qquad p_\psi(v) := \arg\min_u Q_\psi(u, v),$$

where $\gamma_\psi(v)$ is any subgradient in the subdifferential $\partial\psi(v)$ of $\psi(v)$ at the point v. Let $\Phi(\cdot) = \phi(\cdot) + \psi(\cdot)$. For any v, if

$$(2.5) \qquad\qquad \Phi(p_\psi(v)) \leq Q_\psi(p_\psi(v), v),$$

then for any u,

$$(2.6) \qquad\qquad 2\mu(\Phi(u) - \Phi(p_\psi(v))) \geq \|p_\psi(v) - u\|^2 - \|v - u\|^2.$$

Proof. From (2.5), we have

$$(2.7) \qquad \begin{aligned} \Phi(u) - \Phi(p_\psi(v)) \quad &\geq \quad \Phi(u) - Q_\psi(p_\psi(v), v) \\ &= \quad \Phi(u) - \left(\phi(p_\psi(v)) + \psi(v) + \langle \gamma_\psi(v), p_\psi(v) - v \rangle + \frac{1}{2\mu} \|p_\psi(v) - v\|_2^2 \right). \end{aligned}$$

Since ϕ and ψ are convex we have

$$(2.8) \qquad\qquad \phi(u) \geq \phi(p_\psi(v)) + \langle u - p_\psi(v), \gamma_\phi(p_\psi(v)) \rangle,$$

and

(2.9)
$$\psi(u) \geq \psi(v) + \langle u - v, \gamma_\psi(v) \rangle,$$

where $\gamma_\phi(\cdot)$ is a subgradient of $\phi(\cdot)$ and $\gamma_\phi(p_\psi(v))$ satisfies the first-order optimality conditions for (2.4), i.e.,

(2.10)
$$\gamma_\phi(p_\psi(v)) + \gamma_\psi(v) + \frac{1}{\mu}(p_\psi(v) - v) = 0.$$

Summing (2.8) and (2.9) yields

(2.11)
$$\Phi(u) \geq \phi(p_\psi(v)) + \langle u - p_\psi(v), \gamma_\phi(p_\psi(v)) \rangle + \psi(v) + \langle u - v, \gamma_\psi(v) \rangle.$$

Therefore, from (2.7), (2.10) and (2.11) it follows that

$$\Phi(u) - \Phi(p_\psi(v)) \geq \langle \gamma_\psi(v) + \gamma_\phi(p_\psi(v)), u - p_\psi(v) \rangle - \frac{1}{2\mu}\|p_\psi(v) - v\|_2^2$$

(2.12)
$$= \langle -\frac{1}{\mu}(p_\psi(v) - v), u - p_\psi(v) \rangle - \frac{1}{2\mu}\|p_\psi(v) - v\|_2^2$$

$$= \frac{1}{2\mu}\left(\|p_\psi(v) - u\|^2 - \|v - u\|^2\right). \quad \square$$

THEOREM 2.3. *Assume $\nabla f(\cdot)$ is Lipschitz continuous with Lipschitz constant $L(f)$. For $\mu \leq 1/L(f)$, the iterates y^k in Algorithm 4 satisfy*

(2.13)
$$F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{2\mu(k + k_n)}, \quad \forall k,$$

where x^* is an optimal solution of (1.1) and k_n is the number of iterations until the k-th for which $F(x^{k+1}) \leq \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$, i.e., the number of iterations when no skipping step occurs. Thus, the sequence $\{F(y^k)\}$ produced by Algorithm 4 converges to $F(x^*)$. Moreover, if $1/(\beta L(f)) \leq \mu \leq 1/L(f)$ where $\beta \geq 1$, the number of iterations needed to obtain an ϵ-optimal solution is at most $\lceil C/\epsilon \rceil$, where $C = \beta L(f)\|x^0 - x^*\|^2/2$.

Proof. Let I be the set of all iteration indices until $k - 1$-st for which no skipping occurs and let I_c be its complement. Let $I = \{n_i\}$, $i = 0, \ldots, k_n - 1$. It follows that for all $n \in I_c$, $x^{n+1} = y^n$.

For $n \in I$ we can apply Lemma 2.2 to obtain the following inequalities. In (2.6), by letting $\psi = f$, $\phi = g$, $u = x^*$ and $v = x^{n+1}$, we get $p_\psi(v) = y^{n+1}$, $\Phi = F$ and

(2.14)
$$2\mu(F(x^*) - F(y^{n+1})) \geq \|y^{n+1} - x^*\|^2 - \|x^{n+1} - x^*\|^2.$$

Similarly, by letting $\psi = g$, $\phi = f$, $u = x^*$ and $v = y^n$ in (2.6) we get $p_g(v) = x^{n+1}$, $\Phi = F$ and

(2.15)
$$2\mu(F(x^*) - F(x^{n+1})) \geq \|x^{n+1} - x^*\|^2 - \|y^n - x^*\|^2.$$

Taking the summation of (2.14) and (2.15) we get

(2.16)
$$2\mu(2F(x^*) - F(x^{n+1}) - F(y^{n+1})) \geq \|y^{n+1} - x^*\|^2 - \|y^n - x^*\|^2.$$

For $n \in I_c$, (2.14) holds. Then since $x^{n+1} = y^n$ we get

$$(2.17) \qquad 2\mu(F(x^*) - F(y^{n+1})) \geq \|y^{n+1} - x^*\|^2 - \|y^n - x^*\|^2.$$

Summing (2.16) and (2.17) over $n = 0, 1, \ldots, k-1$ we get

$$(2.18) \qquad 2\mu((2|I| + |I_c|)F(x^*) - \sum_{n \in I} F(x^{n+1}) - \sum_{n=0}^{k-1} F(y^{n+1}))$$

$$\geq \sum_{n=0}^{k-1} \left(\|y^{n+1} - x^*\|^2 - \|y^n - x^*\|^2 \right)$$

$$= \|y^k - x^*\|^2 - \|y^0 - x^*\|^2$$

$$\geq - \|x^0 - x^*\|^2.$$

For any n, since Lemma 2.2 holds for any u, letting $u = x^{n+1}$ instead of x^* we get from (2.14) that

$$(2.19) \qquad 2\mu(F(x^{n+1}) - F(y^{n+1})) \geq \|y^{n+1} - x^{n+1}\|^2 \geq 0,$$

or, equivalently,

$$(2.20) \qquad 2\mu(F(x^n) - F(y^n)) \geq \|y^n - x^n\|^2 \geq 0.$$

Thus we get $F(y^n) \leq F(x^n), \forall n$.

Similarly, for $n \in I$ by letting $u = y^n$ instead of x^* we get from (2.15) that

$$(2.21) \qquad 2\mu(F(y^n) - F(x^{n+1})) \geq \|x^{n+1} - y^n\|^2 \geq 0.$$

On the other hand, for $n \in I_c$, (2.21) holds trivially because $x^{n+1} = y^n$; thus (2.21) holds for all n.

Adding (2.19) and (2.21) and adding (2.20) and (2.21), respectively, yield

$$(2.22) \qquad 2\mu(F(y^n) - F(y^{n+1})) \geq 0 \text{ and } 2\mu(F(x^n) - F(x^{n+1})) \geq 0, \text{ for all } n.$$

The inequalities (2.22) show that the sequences of function values $F(y^n)$ and $F(x^n)$ are non-increasing. Thus we have,

$$(2.23) \qquad \sum_{n=0}^{k-1} F(y^{n+1}) \geq kF(y^k) \quad \text{and} \quad \sum_{n \in I} F(x^{n+1}) \geq k_n F(x^k).$$

Combining (2.18) and (2.23) yields

$$(2.24) \qquad 2\mu \left((k + k_n)F(x^*) - k_n F(x^k) - kF(y^k) \right) \geq - \|x^0 - x^*\|^2.$$

Hence, since $F(y^k) \leq F(x^k)$,

$$2\mu(k + k_n) \left(F(y^k) - F(x^*) \right) \leq \|x^0 - x^*\|^2,$$

10

which gives us the desired result (2.13). ☐

COROLLARY 2.4. *Assume* ∇f *and* ∇g *are both Lipschitz continuous with Lipschitz constants* $L(f)$ *and* $L(g)$, *respectively. For* $\mu \leq \min\{1/L(f), 1/L(g)\}$, *Algorithm 3 satisfies*

$$(2.25) \qquad F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{4\mu k}, \quad \forall k,$$

where x^* *is an optimal solution of* (1.1). *Thus sequence* $\{F(y^k)\}$ *produced by Algorithm 3 converges to* $F(x^*)$. *Moreover, if* $1/(\beta \max\{L(f), L(g)\}) \leq \mu \leq 1/\max\{L(f), L(g)\}$ *where* $\beta \geq 1$, *the number of iterations needed to get an* ϵ-*optimal solution is at most* $\lceil C/\epsilon \rceil$, *where* $C = \beta \max\{L(f), L(g)\} \|x^0 - x^*\|^2/4$.

Proof. The conclusion follows from Theorems 2.1 and 2.3 and $k_n = k$. ☐

REMARK 2.5. *The complexity bound in Corollary 2.4 is smaller than the analogous bound for ISTA in [4] by a factor of two. It is easy to see that the bound in Theorem 2.3 is also an improvement over the bound in [4] as long as* $F(x^{k+1}) \leq \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$ *holds for at least one value of* k. *It is reasonable then to ask if the per-iteration cost of Algorithms 3 and 4 are comparable to that of ISTA. It is indeed the case when the assumption holds that minimizing* $\mathcal{L}_\mu(x, y^k; \lambda^k)$ *has comparable cost (and often involves the same computations) as computing the gradient* $\nabla f(y^k)$.

REMARK 2.6. *More general problems of the form*

$$(2.26) \qquad \begin{aligned} \min \quad & f(x) + g(y) \\ s.t. \quad & Ax + y = b \end{aligned}$$

are easily handled by our approach, since one can express (2.26) *as*

$$\min \quad f(x) + g(b - Ax).$$

REMARK 2.7. *If a convex constraint* $x \in \mathcal{C}$, *where* $\mathcal{C}$ *is a convex set is added to problem* (1.1), *and we impose this constraint in the two subproblems in Algorithms 3 and 4, i.e., we impose* $x \in \mathcal{C}$ *in the subproblems with respect to* x *and* $y \in \mathcal{C}$ *in the subproblems with respect to* y, *the complexity results in Theorem 2.3 and Corollary 2.4 continue to hold. The only changes in the proof are in Lemma 2.2. If there is a constraint* $x \in \mathcal{C}$, *then* (2.6) *holds for any* $u \in \mathcal{C}$ *and* $v \in \mathcal{C}$. *Also in the proof of Lemma 2.2, the first equality in* (2.12) *becomes a "*$\geq$*" inequality due to the fact that the optimality conditions* (2.10) *become*

$$\langle \gamma_\phi(p_\psi(v)) + \gamma_\psi(v) + \frac{1}{\mu}(p_\psi(v) - v), u - p_\psi(v) \rangle \geq 0, \forall u \in \mathcal{C}.$$

REMARK 2.8. *Although Algorithms 3 and 4 assume that the Lipschitz constants are known, and hence that an upper bound for* μ *is known, this can be relaxed by using the backtracking technique in [4] to estimate* μ *at each iteration.*

3. Fast Alternating Linearization Methods. In this section, we propose a fast alternating linearization method (FALM) which computes an ϵ-optimal solution to problem (1.1) in $O(\sqrt{L/\epsilon})$ iterations, while keeping the work at each iteration almost the same as that required by ALM.

FALM is an accelerated version of ALM for solving (1.1), or equivalently (1.8), when $f(x)$ and $g(x)$ are both differentiable, and is given below as Algorithm 6. Clearly, FALM is also a Gauss-Seidel type algorithm.

In fact, it is a successive over-relaxation type algorithm since $(t_k - 1)/t_{k+1} > 0, \forall k \geq 2$.

Algorithm 6: Fast Alternating Linearization Method (FALM)

---
1 Choose μ and $x^0 = y^0 = z^1$, set $t_1 = 1$.
2 **for** $k = 1, 2, \cdots$ **do**
3 $\quad$ $x^k := \arg\min_x Q_g(x, z^k)$
4 $\quad$ $y^k := \arg\min_y Q_f(y, x^k)$
5 $\quad$ $t_{k+1} := (1 + \sqrt{1 + 4t_k^2})/2$
6 $\quad$ $z^{k+1} := y^k + \frac{t_k - 1}{t_{k+1}}(y^k - y^{k-1})$

Algorithm 6 requires both f and g to be continuously differentiable. To develop an algorithm that can be applied to problems where one of the functions is non-differentiable, we use a skipping technique as in Algorithm 4. FALM with skipping steps (FALM-S), which does not require $g(x)$ to be smooth, is given below as Algorithm 7.

Algorithm 7: FALM with Skipping Steps (FALM-S)

---
1 Choose $x^0 = y^0 = z^1$ and $\lambda^1 \in -\partial g(z^1)$, set $t_1 = 1$.
2 **for** $k = 1, 2, \cdots$ **do**
3 $\quad$ $x^k := \arg\min_x \mathcal{L}_\mu(x, z^k; \lambda^k)$
4 $\quad$ **if** $F(x^k) > \mathcal{L}_\mu(x^k, z^k; \lambda^k)$ **then**
5 $\quad\quad$ **if** *x-step was not skipped at iteration k-1* **then**
6 $\quad\quad\quad$ $t_k := \left(1 + \sqrt{1 + 8t_{k-1}^2}\right)/2$
7 $\quad\quad$ **else**
8 $\quad\quad\quad$ $t_k := \left(1 + \sqrt{1 + 4t_{k-1}^2}\right)/2$
9 $\quad\quad$ $x^k := z^k := y^{k-1} + \frac{t_{k-1} - 1}{t_k}\left(y^{k-1} - y^{k-2}\right)$
10 $\quad$ $y^k := \arg\min_y Q_f(x^k, y)$
11 $\quad$ **if** $x^k = z^k$ **then**
12 $\quad\quad$ $t_{k+1} := \left(1 + \sqrt{1 + 2t_k^2}\right)/2$
13 $\quad$ **else**
14 $\quad\quad$ $t_{k+1} := \left(1 + \sqrt{1 + 4t_k^2}\right)/2$
15 $\quad$ $z^{k+1} := y^k + \frac{t_k - 1}{t_{k+1}}\left(y^k - y^{k-1}\right)$
16 $\quad$ Choose $\lambda^{k+1} \in -\partial g(z^{k+1})$

The following theorem gives conditions under which Algorithms 6 and 7 are equivalent.

THEOREM 3.1. *If both $f(x)$ and $g(x)$ are differentiable and $\nabla g(x)$ is Lipschitz continuous with Lipschitz constant $L(g)$, and $\mu \leq 1/L(g)$, then Algorithms 6 and 7 are equivalent.*

Proof. As in Theorem 2.1, if f and g are differentiable, $\mathcal{L}_\mu(x, z^k; \lambda^k) \equiv Q_g(x, z^k)$. The conclusion then follows from the fact that $F(x^k) \leq \mathcal{L}_\mu(x^k, z^k; \lambda^k)$ always holds when $\mu \leq 1/L(g)$ and thus there are no skipping steps. $\square$

To prove that Algorithm 7 requires $O(\sqrt{L(f)/\epsilon})$ iterations to obtain an ϵ-optimal solution, we need the following lemmas. We call k-th iteration a *skipping step* if $x^k = z^k$, and a *regular step* if $x^k \neq z^k$.

LEMMA 3.2. *The sequence $\{x^k, y^k\}$ generated by Algorithm 7 satisfies*

$$(3.1) \qquad\qquad 2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \geq \|u^{k+1}\|^2 - \|u^k\|^2,$$

where $u^k := t_k y^k - (t_k - 1)y^{k-1} - x^*$ and $v_k := 2F(y^k) - 2F(x^*)$ if iteration k is a regular step and $v_k := F(y^k) - F(x^*)$ if iteration k is a skipping step.

Proof. There are four cases to consider: (i) both the k-th and the $(k+1)$-st iterations are regular steps; (ii) the k-th iteration is a regular step and the $(k+1)$-st iteration is a skipping step; (iii) both the k-th and the $(k+1)$-st iterations are skipping steps; (iv) the k-th iteration is a skipping step and the $(k+1)$-st iteration is a regular step. We will prove that the following inequality holds for all the four cases:

(3.2)
$$2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1})$$
$$\geq t_{k+1}(t_{k+1} - 1)\left(\|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2\right) + t_{k+1}\left(\|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2\right).$$

The proof of (3.1) and hence, the lemma, then follows from the fact that the right hand side of inequality (3.2) equals

$$\|t_{k+1}y^{k+1} - (t_{k+1} - 1)y^k - x^*\|^2 - \|t_{k+1}z^{k+1} - (t_{k+1} - 1)y^k - x^*\|^2 = \|u^{k+1}\|^2 - \|u^k\|^2,$$

where we have used the fact that $t_{k+1}z^{k+1} := t_{k+1}y^k + t_k(y^k - y^{k-1}) - (y^k - y^{k-1})$.

Case (i): Let us first consider the case when both the k-th and $(k+1)$-st iterations are regular steps. In (2.6), by letting $\psi = f$, $\phi = g$, $u = y^k$ and $v = x^{k+1}$, we get $p_\psi(v) = y^{k+1}$, $\Phi = F$ and

(3.3)
$$2\mu(F(y^k) - F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|x^{k+1} - y^k\|^2.$$

In (2.6), by letting $\psi = g$, $\phi = f$, $u = y^k$, $v = z^{k+1}$, we get $p_\psi(v) = x^{k+1}$, $\Phi = F$ and

(3.4)
$$2\mu(F(y^k) - F(x^{k+1})) \geq \|x^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2.$$

Summing (3.3) and (3.4), and using the fact that $F(y^{k+1}) \leq F(x^{k+1})$, we obtain,

(3.5)
$$2\mu(v_k - v_{k+1}) = 2\mu(2F(y^k) - 2F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2$$

Again, in (2.6), by letting $\psi = g$, $\phi = f$, $u = x^*$, $v = z^{k+1}$, we get $p_\psi(v) = x^{k+1}$, $\Phi = F$ and

(3.6)
$$2\mu(F(x^*) - F(x^{k+1})) \geq \|x^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2.$$

In (2.6), by letting $\psi = f$, $\phi = g$, $u = x^*$, $v = x^{k+1}$, we get $p_\psi(v) = y^{k+1}$, $\Phi = F$ and

(3.7)
$$2\mu(F(x^*) - F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|x^{k+1} - x^*\|^2.$$

Summing (3.6) and (3.7), and again using the fact that $F(y^{k+1}) \leq F(x^{k+1})$, we obtain,

(3.8)
$$-2\mu v_{k+1} = 2\mu(2F(x^*) - 2F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2.$$

If we multiply (3.5) by t_k^2, and (3.8) by t_{k+1}, and take the sum of the resulting two inequalities, we get (3.2) by using the fact that $t_k^2 = t_{k+1}(t_{k+1} - 1)$.

Case (ii): By letting $\psi = f$, $\phi = g$, $u = y^k$ and $v = z^{k+1}$ in (2.6), we get $p_\psi(v) = y^{k+1}$, $\Phi = F$ and

(3.9)
$$2\mu(F(y^k) - F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2.$$

13

Since the steps taken in the k-th and $(k+1)$-st iterations are regular and skipping, respectively, we have

$$(3.10) \qquad 2\mu\left(\frac{v_k}{2} - v_{k+1}\right) = 2\mu(F(y^k) - F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2$$

Also by letting $\psi = f$, $\phi = g$, $u = x^*$ and $v = z^{k+1}$ in (2.6), we get $p_\psi(v) = y^{k+1}$, $\Phi = F$ and

$$(3.11) \qquad -2\mu v_{k+1} = 2\mu(F(x^*) - F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2.$$

Then multiplying (3.10) by $2t_k^2$, (3.11) by t_{k+1}, summing the resulting two inequalities and using the fact that in this case $2t_k^2 = t_{k+1}(t_{k+1} - 1)$, we obtain (3.2).

Case (iii): This case reduces to two consecutive FISTA steps and the proof above applies with $t_k^2 = t_{k+1}(t_{k+1} - 1)$ and inequality (3.10) replaced by

$$(3.12) \qquad 2\mu(v_k - v_{k+1}) = 2\mu(F(y^k) - F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2$$

which gets multiplied by t_k^2.

Case (iv): In this case, (3.5) in the proof of case (i) is replaced by

$$(3.13) \qquad 2\mu(2v_k - v_{k+1}) = 2\mu(2F(y^k) - 2F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2$$

which when multiplied by $t_k^2/2$ and combined with (3.8) multiplied by t_{k+1}, and the fact that in this case $t_k^2/2 = t_{k+1}(t_{k+1} - 1)$, yields (3.2). $\square$

The following lemma gives lower bounds for the sequence of scalars $\{t_k\}$ generated by Algorithm 7.

LEMMA 3.3. *For all $k \geq 1$ the sequence $\{t_k\}$ generated by Algorithm 7 satisfies:*
if the first step is a skipping step,

$$t_k \geq \begin{cases} \frac{1}{2}(k + 1 + \alpha r(k)) & \text{if} \quad k \quad \text{is a skipping step,} \\ \frac{1}{2\sqrt{2}}(k + 1 + \alpha r(k)) & \text{if} \quad k \quad \text{is a regular step,} \end{cases}$$

if the first step is a regular step,

$$t_k \geq \begin{cases} \frac{1}{\sqrt{2}}(k + 1 + \hat{\alpha} s(k)) & \text{if} \quad k \quad \text{is a skipping step,} \\ \frac{1}{2}(k + 1 + \hat{\alpha} s(k)) & \text{if} \quad k \quad \text{is a regular step,} \end{cases}$$

where $r(k)$ and $s(k)$ are the number of steps among the first k steps that are regular and skipping steps, respectively, and $\alpha \equiv \sqrt{2} - 1$ and $\hat{\alpha} \equiv \frac{1}{\sqrt{2}} - 1$.

Proof. Consider the case where the first iteration of Algorithm 7 is a skipping step. Clearly, the sequence of iterations follows a pattern of alternating blocks of one or more skipping steps and one or more regular steps. Let the index of the first iteration in the i-th block be denoted by n_i. Since it is assumed that the first iteration is a skipping step, iterations $n_1, n_3, n_5 \ldots$ are skipping steps ($n_1 = 1$) and $n_2, n_4, n_6 \ldots$ are regular steps. Note that the statement of the lemma in this case corresponds to

$$(3.14) \qquad t_k \geq \begin{cases} \frac{1}{2}(k + 1 + \alpha r(k)), & \text{for } n_j \leq k \leq n_{j+1} - 1, \text{ if } j \text{ is odd,} \\ \frac{1}{2\sqrt{2}}(k + 1 + \alpha r(k)), & \text{for } n_j \leq k \leq n_{j+1} - 1, \text{ if } j \text{ is even,} \end{cases}$$

which we will prove by induction on j.

14

We first note that it follows from the updating rules and formulas for t_k that

$$(3.15) \qquad t_k \geq \begin{cases} \frac{1}{2} + \sqrt{2}t_{k-1}, & \text{if } k \text{ is a skipping step and } k-1 \text{ is a regular step,} \\ \frac{1}{2} + \frac{1}{\sqrt{2}}t_{k-1}, & \text{if } k \text{ is a regular step and } k-1 \text{ is a skipping step,} \\ \frac{1}{2} + t_{k-1}, & \text{otherwise.} \end{cases}$$

Consider $j = 1$. Clearly (3.14) holds for all iterations $n_1 = 1 \leq k \leq n_2 - 1$, since $t_k \geq \frac{k+1}{2}$ holds trivially for $t_1 = 1$, and for $1 < k \leq n_2 - 1$, $t_k \geq \frac{k-1}{2} + t_1 = \frac{k+1}{2}$.

Now assume that (3.14) holds for all $j < \bar{j}$. If $\bar{j}$ is even, iterations $k = n_{\bar{j}}$ and $k - 1$ are, respectively, regular and skipping iterations. Hence, from (3.15), we have that

$$t_k \geq \frac{1}{2} + \frac{1}{\sqrt{2}}t_{k-1} \geq \frac{1}{2} + \frac{1}{2\sqrt{2}}(k + \alpha r(k-1)) = \frac{1}{2\sqrt{2}}(k + 1 + \alpha r(k)).$$

Since the remaining $p \equiv n_{\bar{j}+1} - 1 - n_{\bar{j}}$ iterations before iteration $n_{\bar{j}+1}$ are all regular iterations (p may be zero), we have from (3.15) that, for $n_{\bar{j}} < k \leq n_{\bar{j}+1} - 1$,

$$\begin{aligned} t_k &\geq \frac{k-n_{\bar{j}}}{2} + t_{n_{\bar{j}}} \geq \frac{k-n_{\bar{j}}}{2} + \frac{1}{2\sqrt{2}}(n_{\bar{j}} + 1 + \alpha r(n_{\bar{j}})) \\ &= \frac{k-n_{\bar{j}}}{2} + \frac{1}{2\sqrt{2}}(n_{\bar{j}} + 1 + \alpha(r(k) - k + n_{\bar{j}})) = \frac{1}{2\sqrt{2}}(k + 1 + \alpha r(k)). \end{aligned}$$

If $\bar{j}$ is odd, iteration $k = n_{\bar{j}}$ is a skipping iteration. Hence, from (3.15), we have that $t_k \geq \frac{1}{2} + \sqrt{2}t_{k-1} \geq \frac{1}{2} + \sqrt{2}\frac{1}{2\sqrt{2}}(k + \alpha r(k-1)) = \frac{1}{2}(k + 1 + \alpha r(k))$. Since the remaining $p \equiv n_{\bar{j}+1} - 1 - n_{\bar{j}}$ iterations before iteration $n_{\bar{j}+1}$ are all skipping iterations (again p may be zero), we have from (3.15) that, for $n_{\bar{j}} < k \leq n_{\bar{j}+1} - 1$, $t_k \geq \frac{k-n_{\bar{j}}}{2} + t_{n_{\bar{j}}} \geq \frac{k-n_{\bar{j}}}{2} + \frac{1}{2}(n_{\bar{j}} + 1 + \alpha r(n_{\bar{j}})) = \frac{1}{2}(k + 1 + \alpha r(k))$. This concludes the induction.

Since the proof for the case that the first step is a regular step is totally analogous to the above proof, we leave this to the reader. $\square$

Now we are ready to give the complexity of Algorithm 7.

THEOREM 3.4. *Let* $\alpha = \sqrt{2} - 1$ *and* $r(k)$ *be the number of steps among the first* k *steps that are regular steps. Assuming* $\nabla f(\cdot)$ *is Lipschitz continuous with Lipschitz constant* $L(f)$, *if* $\mu \leq 1/L(f)$, *the sequence* $\{y^k\}$ *generated by Algorithm 7 satisfies:*

$$(3.16) \qquad F(y^k) - F(x^*) \leq \frac{2\|x^0 - x^*\|^2}{\mu(k + 1 + \alpha\hat{r}(k))^2},$$

where $\hat{r}(k) = r(k)$ *if the first step is a skipping step, and* $\hat{r}(k) = r(k) + 1$ *if the first step is a regular step.*

Hence, the sequence $\{F(y^k)\}$ *produced by Algorithm 4 converges to* $F(x^*)$. *Moreover, if* $1/(\beta L(f)) \leq \mu \leq 1/L(f)$ *where* $\beta \geq 1$, *the number of iterations required by Algorithm 7 to get an* ϵ-*optimal solution to* (1.1) *is at most* $\lfloor\sqrt{C/\epsilon}\rfloor$, *where* $C = 2\beta L(f)\|x^0 - x^*\|^2$.

Proof. Using the same notation as in Lemmas 3.2 and 3.3, (3.8) and (3.11) imply that

$$-2\mu v_1 \geq \|y^1 - x^*\|^2 - \|z^1 - x^*\|^2$$

holds whether the first iteration is a skipping step or not. Thus we have

$$(3.17) \qquad 2\mu v_1 + \|y^1 - x^*\|^2 \leq \|z^1 - x^*\|^2 = \|x^0 - x^*\|^2.$$

15

From Lemma 3.2 we know that the sequence $\{2\mu t_k^2 v_k + \|u^k\|^2\}$ is non-increasing. Therefore, we have

$$(3.18) \qquad 2\mu t_k^2 v_k \leq 2\mu t_k^2 v_k + \|u^k\|^2 \leq 2\mu t_1^2 v_1 + \|u^1\|^2 = 2\mu v_1 + \|y^1 - x^*\|^2 \leq \|x^0 - x^*\|^2,$$

where the equality follows from the facts that $t_1 = 1$ and $u^1 = y^1 - x^*$, and the last inequality is from (3.17).

Recall the definition of v_k in Lemma 3.2 and the bounds of t_k in Lemma 3.3. We get (i) and (ii) from (3.18). Keeping in mind that v_k has a different expression depending on whether the k-th step is a skipping or a regular step, it follows that the sequence $\{y^k\}$ generated by Algorithm 7 satisfies:

(i) if the first step is a skipping step, then

$$F(y^k) - F(x^*) \leq \frac{2\|x^0 - x^*\|^2}{\mu(k + 1 + \alpha r(k))^2};$$

(ii) if the first step is a regular step, then

$$F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{\mu(k + 1 + \hat{\alpha}s(k))^2}.$$

It is easy to check that these bounds are equivalent to (3.16), and that the worst case bound on the number of iterations follows from (3.16). $\square$

COROLLARY 3.5. *Assume ∇f and ∇g are both Lipschitz continuous with Lipschitz constants $L(f)$ and $L(g)$, respectively. For $\mu \leq \min\{1/L(f), 1/L(g)\}$, Algorithm 6 satisfies*

$$(3.19) \qquad F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{\mu(k + 1)^2}, \quad \forall k,$$

where x^* is an optimal solution of (1.1). Hence, the sequence $\{F(y^k)\}$ produced by Algorithm 6 converges to $F(x^*)$, and if $1/(\beta \max\{L(f), L(g)\}) \leq \mu \leq 1/\max\{L(f), L(g)\}$, where $\beta \geq 1$, the number of iterations needed to get an ϵ-optimal solution is at most $\lceil \sqrt{C/\epsilon} - 1 \rceil$, where $C = \beta \max\{L(f), L(g)\}\|x^0 - x^*\|^2$.

Proof. Note that since $\mu \leq \min\{1/L(f), 1/L(g)\}$, from Theorem 3.1 we know that Algorithms 6 and 7 are equivalent. That is, every step in Algorithm 7 is a regular step. Therefore, case (ii) in Theorem 3.4 holds and $s(k) = 0$, which leads to (3.19). $\square$

REMARK 3.6. *The complexity bound in Corollary 3.5 is smaller than the analogous bound for FISTA in [4] by a factor of $\sqrt{2}$. It is easy to see that the bound in Theorem 3.4 is also an improvement over the bound in [4] as long as $F(x^{k+1}) \leq \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$ holds for at least one value of k. As in the case of Algorithms 3 and 4, the per-iteration cost of Algorithms 6 and 7 are comparable to that of FISTA.*

REMARK 3.7. *Line 6 in Algorithm 6 and Line 15 in Algorithm 7 can be changed to:*

$$z^{k+1} := w^k + \frac{1}{t_{k+1}}[t_k(y^k - w^{k-1}) - (w^k - w^{k-1})],$$

where $w^k := \alpha x^k + (1 - \alpha)y^k, \alpha \in (0, 1)$, and Theorem 3.4 and Corollary 3.5 still hold.

REMARK 3.8. *Although Algorithms 6 and 7, assume that the Lipschitz constants are known, and hence that an upper bound for μ is known, this can be relaxed by using the backtracking technique in [4] to estimate μ at each iteration.*

4. Comparison of ALM, FALM, ISTA, FISTA, SADAL and SALSA. In this section we compare the performance of our basic and fast ALMs, with and without skipping steps, against ISTA, FISTA,

SADAL (Algorithm 2) and an alternating direction augmented Lagrangian method SALSA described in [2] on a benchmark wavelet-based image deblurring problem from [17]. In this problem, the original image is the well-known Cameraman image of size 256×256 and the observed image is obtained after imposing a uniform blur of size 9×9 (denoted by the operator R) and Gaussian noise (generated by the function *randn* in MATLAB with a seed of 0 and a standard deviation of 0.56). Since the coefficient of the wavelet transform of the image is sparse in this problem, one can try to reconstruct the image u from the observed image b by solving the problem:

$$(4.1) \qquad \bar{x} := \arg\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1,$$

and setting $u := W\bar{x}$, where $A := RW$ and W is the inverse discrete Haar wavelet transform with four levels. By defining $f(x) := \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) := \rho\|x\|_1$, it is clear that (4.1) can be expressed in the form of (1.1) and can be solved by ALM-S (Algorithm 4), FALM-S (Algorithm 7), ISTA, FISTA, SALSA and SADAL (Algorithm 2). However, in order to use ALM (Algorithm 3) and FALM (Algorithm 6), we need to smooth $g(x)$ first, since these two algorithms require both f and g to be smooth. Here we apply the smoothing technique introduced by Nesterov [39] since this technique guarantees that the gradient of the smoothed function is Lipschitz continuous. A smoothed approximation to the ℓ_1 function $g(x) := \rho\|x\|_1$ with smoothness parameter $\sigma > 0$ is

$$(4.2) \qquad g_\sigma(x) := \max\{\langle x, z\rangle - \frac{\sigma}{2}\|z\|_2^2 : \|z\|_\infty \le \rho\}.$$

It is easy to show that the optimal solution $z_\sigma(x)$ of (4.2) is

$$(4.3) \qquad z_\sigma(x) = \min\{\rho, \max\{x/\sigma, -\rho\}\}.$$

According to Theorem 1 in [39], the gradient of g_σ is given by $\nabla g_\sigma(x) = z_\sigma(x)$ and is Lipschitz continuous with Lipschitz constant $L(g_\sigma) = 1/\sigma$. After smoothing g, we can apply Algorithms 3 and 6 to solve the smoothed problem:

$$(4.4) \qquad \min_x f(x) + g_\sigma(x).$$

We have the following theorem about the ϵ-optimal solutions of problems (4.1) and (4.4).

THEOREM 4.1. *Let $\sigma = \frac{\epsilon}{n\rho^2}$ and $\epsilon > 0$. If $x(\sigma)$ is an $\epsilon/2$-optimal solution to (4.4), then $x(\sigma)$ is an ϵ-optimal solution to (4.1).*

Proof. Let $D_g := \max\{\frac{1}{2}\|z\|_2^2 : \|z\|_\infty \le \rho\} = \frac{1}{2}n\rho^2$ and x^* and $x^*(\sigma)$ be optimal solution to problems (4.1) and (4.4), respectively. Note that

$$(4.5) \qquad g_\sigma(x) \le g(x) \le g_\sigma(x) + \sigma D_g, \forall x \in \mathbb{R}^n.$$

Using the inequalities in (4.5) and the facts that $x(\sigma)$ is an $\epsilon/2$-optimal solution to (4.4) and $\sigma D_g = \frac{\epsilon}{2}$, we

have

$$f(x(\sigma)) + g(x(\sigma)) - f(x^*) - g(x^*) \le f(x(\sigma)) + g_\sigma(x(\sigma)) + \sigma D_g - f(x^*) - g_\sigma(x^*)$$
$$\le f(x(\sigma)) + g_\sigma(x(\sigma)) + \sigma D_g - f(x^*(\sigma)) - g_\sigma(x^*(\sigma))$$
$$\le \epsilon/2 + \sigma D_g = \epsilon. \quad \square$$

Thus, to find an ϵ-optimal solution to (4.1), we can apply Algorithms 3 and 6 to find an $\epsilon/2$-optimal solution to (4.4) with $\sigma = \frac{\epsilon}{n\rho^2}$. The iteration complexity results in Corollaries 2.4 and 3.5 hold since the gradient of g_σ is Lipschitz continuous. However, the numbers of iterations needed by ALM and FALM to obtain an ϵ-optimal solution to (4.1) become $O(1/\epsilon^2)$ and $O(1/\epsilon)$, respectively, due to the fact that the Lipschitz constant $L(g_\sigma) = 1/\sigma = \frac{n\rho^2}{\epsilon} = O(1/\epsilon)$.

When Algorithms 4 and 7 are applied to solve (4.1), the subproblems (1.2) and (1.3) are easy to solve. Specifically, (1.2) corresponds to solving a linear system which is particularly easy to do because of the special structures of R and W (see [1, 2]); (1.3) corresponds to a vector shrinkage operation. When Algorithms 3 and 6 are applied to solve (4.4), (1.3) with g replaced by g_σ is also easy to solve; its optimal solution is

$$x := z - \tau \min\{\rho, \max\{-\rho, \frac{z}{\tau + \sigma}\}\}.$$

Since ALM is equivalent to SADAL when both functions are smooth, we implemented ALM as SADAL when we solved (4.4). We also applied SADAL to the nonsmooth problem (4.1). We also implemented ALM-S as Algorithm 5 since the latter was usually faster. In all algorithms, we set the initial points $x^0 = y^0 = \mathbf{0}$, and in FALM and FALM-S we set $z^1 = \mathbf{0}$. MATLAB codes for SALSA, FISTA and ISTA (modified from FISTA) were downloaded from http://cascais.lx.it.pt/~mafonso/salsa.html and their default inputs were used. Moreover, λ^0 was set to $\mathbf{0}$ in algorithms 2, 4, 5 and 7 since $\nabla g_\sigma(x^0) = \mathbf{0}$ and $\mathbf{0} \in -\partial g(x^0)$ when $x^0 = \mathbf{0}$. Also, whenever $g(x)$ was smoothed, we set $\sigma = 10^{-6}$. μ was set to 1 in all the algorithms since the Lipschitz constant of the gradient of function $\frac{1}{2}\|RW(\cdot) - b\|_2^2$ was known to be 1. We set μ to 1 even for the smoothed problems. Although this violates the requirement $\mu \le \frac{1}{L(g_\sigma)}$ in Corollaries 2.4 and 3.5, we see from our numerical results reported below that ALM and FALM still work very well. All of the algorithms tested were terminated after 1000 iterations. The (nonsmoothed) objective function values in (4.1) produced by these algorithms at iterations: 10, 50, 100, 200, 500, 800 and 1000 for different choices of ρ are presented in Tables 4.1 and 4.2. The CPU times (in seconds) and the number of iterations required to reduce the objective function value to below $1.04e + 5$ and $8.60e + 5$ are reported respectively in the last columns of Tables 4.1 and 4.2.

All of our codes were written in MATLAB and run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

TABLE 4.1
Comparison of the algorithms for solving (4.1) *with* $\rho = 0.01$

solver	obj in k-th iteration							cpu (iter)
	10	50	100	200	500	800	1000	
FALM-S	1.767239e+5	1.040919e+5	1.004322e+5	9.726599e+4	9.341282e+4	9.182962e+4	9.121742e+4	24.3 (51)
FALM	1.767249e+5	1.040955e+5	9.899843e+4	9.516208e+4	9.186355e+4	9.073086e+4	9.028790e+4	23.1 (51)
FISTA	1.723109e+5	1.061116e+5	1.016385e+5	9.752858e+4	9.372093e+4	9.233719e+4	9.178455e+4	26.0 (69)
ALM-S	4.218082e+5	1.439742e+5	1.212865e+5	1.107103e+5	1.042869e+5	1.021905e+5	1.013128e+5	208.9 (531)
ALM	4.585705e+5	1.481379e+5	1.233182e+5	1.116683e+5	1.047410e+5	1.025611e+5	1.016589e+5	208.1 (581)
ISTA	2.345290e+5	1.267048e+5	1.137827e+5	1.079721e+5	1.040666e+5	1.025107e+5	1.018068e+5	196.8 (510)
SALSA	8.772957e+5	1.549462e+5	1.267379e+5	1.132676e+5	1.054600e+5	1.031346e+5	1.021898e+5	223.9 (663)
SADAL	2.524912e+5	1.271591e+5	1.133542e+5	1.068386e+5	1.021905e+5	1.004005e+5	9.961905e+4	113.5 (332)

solver	obj in k-th iteration							cpu (iter)
	10	50	100	200	500	800	1000	
FALM-S	9.868574e+5	8.771604e+5	8.487372e+5	8.271496e+5	8.110211e+5	8.065750e+5	8.050973e+5	37.7 (76)
FALM	9.876315e+5	8.629257e+5	8.369244e+5	8.210375e+5	8.097621e+5	8.067903e+5	8.058290e+5	25.7 (54)
FISTA	9.924884e+5	8.830263e+5	8.501727e+5	8.288459e+5	8.126598e+5	8.081259e+5	8.066060e+5	30.1 (79)
ALM-S	1.227588e+6	9.468694e+5	9.134766e+5	8.880703e+5	8.617264e+5	8.509737e+5	8.465260e+5	214.5 (537)
ALM	1.263787e+6	9.521381e+5	9.172737e+5	8.910902e+5	8.639917e+5	8.528932e+5	8.482666e+5	211.8 (588)
ISTA	1.048956e+6	9.396822e+5	9.161787e+5	8.951970e+5	8.700864e+5	8.589587e+5	8.541664e+5	293.8 (764)
SALSA	1.680608e+6	9.601661e+5	9.230268e+5	8.956607e+5	8.674579e+5	8.558580e+5	8.509770e+5	230.2 (671)
SADAL	1.060130e+6	9.231803e+5	8.956150e+5	8.735746e+5	8.509601e+5	8.420295e+5	8.383270e+5	112.5 (335)

From Tables 4.1 and 4.2 we see that in terms of the value of the objective function achieved after a specified number of iterations, the performance of FALM-S and FALM is always slightly better than that of FISTA and is much better than the performance of the other algorithms. On the two test problems, since FALM-S and FALM are always better than ALM-S and ALM, and FISTA is always better than ISTA, we can conclude that the Nesterov-type acceleration technique greatly speeds up the basic algorithms on these problems. Moreover, although sometimes in the early iterations FISTA (ISTA) is better than FALM-S and FALM (ALM-S and ALM), it is always worse than the latter two algorithms when the iteration number is large. We also illustrate our comparisons graphically by plotting in Figure 4.1 the objective function value versus the number of iterations taken by these algorithms for solving (4.1) with $\rho = 0.1$. From Figure 4.1 we see clearly that for this problem, ALM outperforms ISTA, FALM outperforms FISTA, FALM outperforms ALM and FALM-S outperforms ALM-S.

From the CPU times and the iteration numbers in the last columns of Tables 4.1 and 4.2 we see that, the fast versions are always much better than the basic versions of the algorithms. Since iterations of FISTA cost less than those of FALM-S (and FALM as well), we see that although FISTA takes 35% (4%) more iterations than FALM-S in the last column of Table 4.1 (4.2) it takes only 7% more time (20% less time).

We note that for the problems with $\rho = 0.01$ and $\rho = 0.1$, (i.e., for the results given in Tables 4.1 and 4.2), 891 and 981, respectively, of the first 1000 iterations performed by FALM-S were skipping steps. In contrast, none of the steps performed by ALM-S on either of these problems were skipping steps. While the latter result is somewhat surprising, the fact that FALM-S performs many skipping steps is not, since the Nesterov-like acceleration approach is an *over-relaxation* approach that generates points that extrapolate beyond the previous point and the one produced by the ALM algorithm.

5. Applications. In this section, we describe how ALM and FALM can be applied to problems that can be formulated as RPCA problems to illustrate the use of Nesterov-type smoothing when the functions f and g do not satisfy the smoothness conditions required by the theorems in Sections 2 and 3. Our numerical results show that our methods are able to solve huge problems that arise in practice; e.g., one problem involving roughly 40 million variables and 20 million linear constraints is solved in about three-quarters of an hour. We alse describe application of our methods to the SICS problem.

5.1. Applications in Robust Principal Component Analysis. In order to apply Algorithms 3 and 6 to (1.6), we need to smooth both the nuclear norm $f(X) := \|X\|_*$ and the ℓ_1 norm $g(Y) := \rho\|Y\|_1$. We again apply Nesterov's smoothing technique as in Section 4. $g(Y)$ can be smoothed in the same way as the vector ℓ_1 norm in Section 4. We use $g_\sigma(Y)$ to denote the smoothed function with smoothness parameter $\sigma > 0$. A smoothed approximation to $f(X)$ with smoothness parameter $\sigma > 0$ is

$$(5.1) \qquad f_\sigma(X) := \max\{\langle X, W \rangle - \frac{\sigma}{2}\|W\|_F^2 : \|W\| \le 1\}.$$
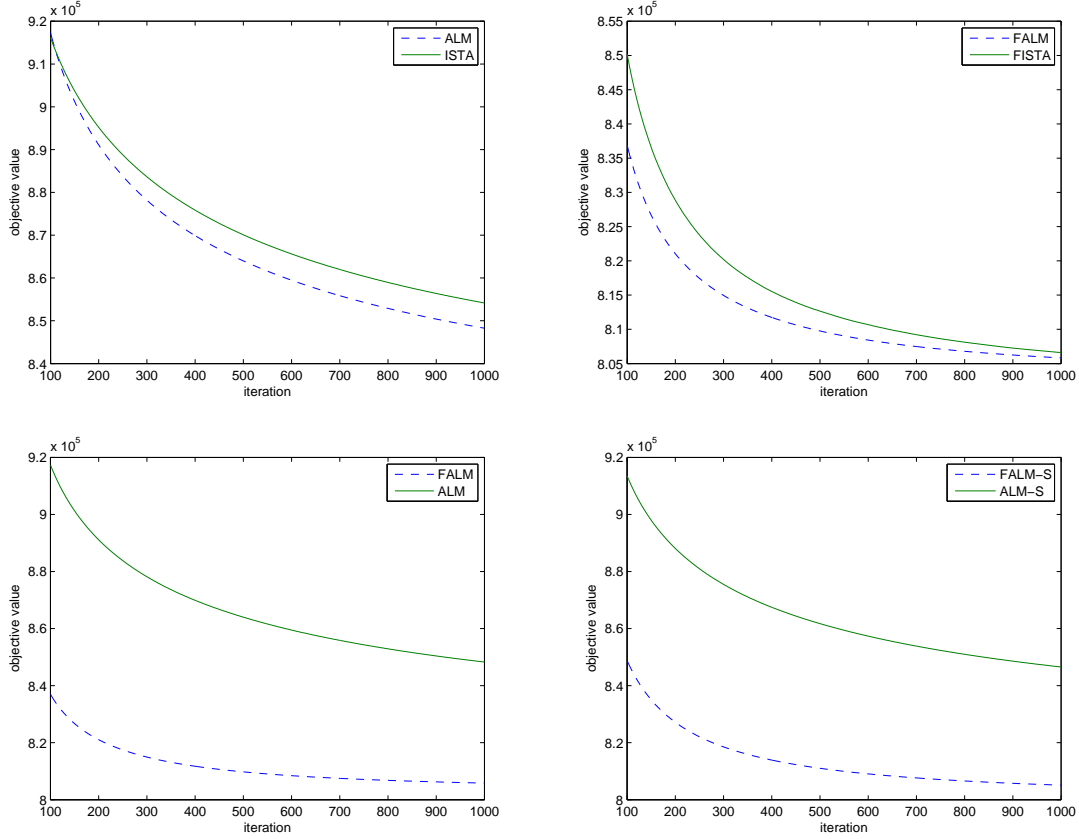
FIG. 4.1. *comparison of the algorithms*

It is easy to show that the optimal solution $W_\sigma(X)$ of (5.1) is

$$(5.2) \qquad\qquad W_\sigma(X) = U\text{Diag}(\min\{\gamma, 1\})V^\top,$$

where $U\text{Diag}(\gamma)V^\top$ is the singular value decomposition (SVD) of X/σ. According to Theorem 1 in [39], the gradient of f_σ is given by $\nabla f_\sigma(X) = W_\sigma(X)$ and is Lipschitz continuous with Lipschitz constant $L(f_\sigma) = 1/\sigma$. After smoothing f and g, we can apply Algorithms 3 and 6 to solve the following smoothed problem:

$$(5.3) \qquad\qquad \min\{f_\sigma(X) + g_\sigma(Y) : X + Y = M\}.$$

We have the following theorem about ϵ-optimal solutions of problems (1.6) and (5.3).

THEOREM 5.1. *Let* $\sigma = \frac{\epsilon}{2\max\{\min\{m,n\}, mn\rho^2\}}$ *and* $\epsilon > 0$. *If* $(X(\sigma), Y(\sigma))$ *is an* $\epsilon/2$-*optimal solution to* (5.3), *then* $(X(\sigma), Y(\sigma))$ *is an* ϵ-*optimal solution to* (1.6).

Proof. Let $D_f := \max\{\frac{1}{2}\|W\|_F^2 : \|W\| \leq 1\} = \frac{1}{2}\min\{m,n\}$, $D_g := \max\{\frac{1}{2}\|Z\|_F^2 : \|Z\|_\infty \leq \rho\} = \frac{1}{2}mn\rho^2$ and (X^*, Y^*) and $(X^*(\sigma), Y^*(\sigma))$ be optimal solution to problems (1.6) and (5.3), respectively. Note that

$$(5.4) \qquad\qquad f_\sigma(X) \leq f(X) \leq f_\sigma(X) + \sigma D_f, \forall X \in \mathbb{R}^{m\times n}$$

and

$$(5.5) \qquad\qquad g_\sigma(Y) \le g(Y) \le g_\sigma(Y) + \sigma D_g, \forall Y \in \mathbb{R}^{m \times n}.$$

Using the inequalities in (5.4) and (5.5) and the facts that $(X(\sigma), Y(\sigma))$ is an $\epsilon/2$-optimal solution to (5.3) and $\sigma \max\{D_f, D_g\} = \frac{\epsilon}{4}$, we have

$$
\begin{aligned}
f(X(\sigma)) + g(Y(\sigma)) - f(X^*) - g(Y^*) &\le f_\sigma(X(\sigma)) + g_\sigma(Y(\sigma)) + \sigma D_f + \sigma D_g - f_\sigma(X^*) - g_\sigma(Y^*) \\
&\le f_\sigma(X(\sigma)) + g_\sigma(Y(\sigma)) + \sigma D_f + \sigma D_g - f_\sigma(X^*(\sigma)) - g_\sigma(Y^*(\sigma)) \\
&\le \epsilon/2 + \sigma D_f + \sigma D_g \le \epsilon/2 + \epsilon/4 + \epsilon/4 = \epsilon. \qquad \square
\end{aligned}
$$

Thus, according to Theorem 5.1, to find an ϵ-optimal solution to (1.6), we need to find an $\epsilon/2$-optimal solution to (5.3) with $\sigma = \frac{\epsilon}{2\max\{\min\{m,n\}, mn\rho^2\}}$. We can either apply Algorithms 3 and 6 to solve (5.3), or apply Algorithms 4 and 7 to solve (5.3) with only one functions (say $f(x)$) smoothed. The iteration complexity results in Theorems 2.3 and 3.4 hold since the gradients of f_σ is Lipschitz continuous. However, the numbers of iterations needed by ALM and FALM to obtain an ϵ-optimal solution to (1.6) become $O(1/\epsilon^2)$ and $O(1/\epsilon)$, respectively, due to the fact that the Lipschitz constant $L(f_\sigma) = 1/\sigma = \frac{2\max\{\min\{m,n\}, mn\rho^2\}}{\epsilon} = O(1/\epsilon)$.

The two subproblems at iteration k of Algorithm 3 when applied to (5.3) reduce to

$$(5.6) \qquad X^{k+1} := \arg\min_X f_\sigma(X) + g_\sigma(Y^k) + \langle \nabla g_\sigma(Y^k), M - X - Y^k \rangle + \frac{1}{2\mu}\|X + Y^k - M\|_F^2,$$

and

$$(5.7) \quad Y^{k+1} := \arg\min_Y f_\sigma(X^{k+1}) + \langle \nabla f_\sigma(X^{k+1}), M - X^{k+1} - Y \rangle + \frac{1}{2\mu}\|X^{k+1} + Y - M\|_F^2 + g_\sigma(Y).$$

The first-order optimality conditions for (5.6) are:

$$(5.8) \qquad\qquad W_\sigma(X) - Z_\sigma(Y^k) + \frac{1}{\mu}(X + Y^k - M) = 0,$$

where $W_\sigma(X)$ and $Z_\sigma(Y)$ are defined in (5.2) and (4.3). It is easy to check that

$$(5.9) \qquad\qquad X := U\mathrm{Diag}(\gamma - \frac{\mu\gamma}{\max\{\gamma, \mu + \sigma\}})V^\top$$

satisfies (5.8), where $U\mathrm{Diag}(\gamma)V^\top$ is the SVD of the matrix $\mu Z_\sigma(Y^k) - Y^k + M$. Thus, solving the subproblem (5.6) corresponds to an SVD. If we define $B := \mu W_\sigma(X^{k+1}) - X^{k+1} + M$, it is easy to verify that

$$(5.10) \qquad Y_{ij} = B_{ij} - \mu\min\{\rho, \max\{-\rho, \frac{B_{ij}}{\sigma + \mu}\}\} \text{ for } i = 1, \dots, m \text{ and } j = 1, \dots, n$$

satisfies the first-order optimality conditions for (5.7): $-W_\sigma(X^{k+1}) + \frac{1}{\mu}(X^{k+1} + Y - M) + Z_\sigma(Y) = 0$. Thus, solving the subproblem (5.7) can be done very cheaply. The two subproblems at the k-th iteration of Algorithm 6 can be done in the same way and the main computational effort in each iteration of both ALM and FALM corresponds to an SVD.

5.2. RPCA with Missing Data. In some applications of RPCA, some of the entries of M in (1.6) may be missing (e.g., in low-rank matrix completion problems where the matrix is corrupted by noise). Let Ω be the index set of the entries of M that are observable and define the projection operator $\mathcal{P}_\Omega$ as: $(\mathcal{P}_\Omega(X))_{ij} = X_{ij}$, if $(i,j) \in \Omega$ and $(\mathcal{P}_\Omega(X))_{ij} = 0$ otherwise. It has been shown under some randomness hypotheses that the low rank $\bar{X}$ and sparse $\bar{Y}$ can be recovered with high probability by solving (see Theorem 1.2 in [6]),

$$(5.11) \qquad (\bar{X}, \bar{Y}) := \arg\min_{X,Y}\{\|X\|_* + \rho\|Y\|_1 : \mathcal{P}_\Omega(X + Y) = \mathcal{P}_\Omega(M)\}.$$

To solve (5.11) by ALM or FALM, we need to transform it into the form of (1.6). For this we have

THEOREM 5.2. $(\bar{X}, \mathcal{P}_\Omega(\bar{Y}))$ *is an optimal solution to* (5.11) *if*

$$(5.12) \qquad (\bar{X}, \bar{Y}) = \arg\min_{X,Y}\{\|X\|_* + \rho\|\mathcal{P}_\Omega(Y)\|_1 : X + Y = \mathcal{P}_\Omega(M)\}.$$

Proof. Suppose (X^*, Y^*) is an optimal solution to (5.11). We claim that $Y^*_{ij} = 0, \forall (i,j) \notin \Omega$. Otherwise, $(X^*, \mathcal{P}_\Omega(Y^*))$ is feasible to (5.11) and has a strictly smaller objective function value than (X^*, Y^*), which contradicts the optimality of (X^*, Y^*). Thus, $\|\mathcal{P}_\Omega(Y^*)\|_1 = \|Y^*\|_1$. Now suppose that $(\bar{X}, \mathcal{P}_\Omega(\bar{Y}))$ is not optimal to (5.11); then we have

$$(5.13) \qquad \|X^*\|_* + \rho\|\mathcal{P}_\Omega(Y^*)\|_1 = \|X^*\|_* + \rho\|Y^*\|_1 < \|\bar{X}\|_* + \rho\|\mathcal{P}_\Omega(\bar{Y})\|_1.$$

By defining a new matrix $\tilde{Y}$ as

$$\tilde{Y}_{ij} = \begin{cases} Y^*_{ij}, & (i,j) \in \Omega \\ -X^*_{ij}, & (i,j) \notin \Omega, \end{cases}$$

we have that $(X^*, \tilde{Y})$ is feasible to (5.12) and $\|\mathcal{P}_\Omega(\tilde{Y})\|_1 = \|\mathcal{P}_\Omega(Y^*)\|_1$. Combining this with (5.13), we obtain

$$\|X^*\|_* + \rho\|\mathcal{P}_\Omega(\tilde{Y})\|_1 < \|\bar{X}\|_* + \rho\|\mathcal{P}_\Omega(\bar{Y})\|_1,$$

which contradicts the optimality of $(\bar{X}, \bar{Y})$ to (5.12). Therefore, $(\bar{X}, \mathcal{P}_\Omega(\bar{Y}))$ is optimal to (5.11). $\square$

The only differences between (1.6) and (5.12) lie in that the matrix M is replaced by $\mathcal{P}_\Omega(M)$ and $g(Y) = \rho\|Y\|_1$ is replaced by $\rho\|\mathcal{P}_\Omega(Y)\|_1$. A smoothed approximation $g_\sigma(Y)$ to $g(Y) := \rho\|\mathcal{P}_\Omega(Y)\|_1$ is given by

$$(5.14) \qquad g_\sigma(Y) := \max\{\langle \mathcal{P}_\Omega(Y), Z \rangle - \frac{\sigma}{2}\|Z\|_F^2 : \|Z\|_\infty \le \rho\},$$

and

$$(5.15) \qquad (\nabla g_\sigma(Y))_{ij} = \min\{\rho, \max\{(\mathcal{P}_\Omega(Y))_{ij}/\sigma, -\rho\}\}, \text{ for } 1 \le i \le m \text{ and } 1 \le j \le n.$$

According to Theorem 1 in [39], $\nabla g_\sigma(Y)$ is Lipschitz continuous with $L_\sigma(g) = 1/\sigma$. Thus the convergence and iteration complexity results in Theorems 2.3 and 3.4 apply. The only changes in Algorithms 3 and 4

and Algorithms 6 and 7 are: replacing M by $\mathcal{P}_\Omega(M)$ and computing Y^{k+1} using (5.10) with B is replaced by $\mathcal{P}_\Omega(B)$.

5.3. Numerical Results on RPCA Problems. In this section, we report numerical results obtained using the ALM method to solve RPCA problems with both complete and incomplete data matrices M. We compare the performance of ALM with the exact ADM (EADM) and the inexact ADM (IADM) methods in [31]. The MATLAB codes of EADM and IADM were downloaded from $http://watt.csl.illinois.edu/\sim$ $perceive/matrix-rank/sample\_code.html$ and their default settings were used. To further accelerate ALM, we adopted the continuation strategy used in EADM and IADM. Specifically, we set $\mu_{k+1} := \max\{\bar{\mu}, \eta\mu_k\}$, where $\mu_0 = \|M\|/1.25, \bar{\mu} = 10^{-6}$ and $\eta = 2/3$ in our numerical experiments. Although in some iterations this violates the requirement $\mu \le \min\{\frac{1}{L(f_\sigma)}, \frac{1}{L(g_\sigma)}\}$ in Corollaries 2.4 and 3.5, we see from our numerical results reported below that ALM and FALM still work very well. We also found that by adopting this updating rule for μ, there was not much difference between the performance of ALM and that of FALM. So we only compare ALM with EADM and IADM. As in Section 4, since we applied ALM to a smoothed problem, we implemented ALM as SADAL. The initial point in ALM was set to $(X^0, Y^0) = (M, \mathbf{0})$ and the initial Lagrange multiplier was set to $\Lambda^0 = -\nabla g_\sigma(Y^0)$. We set the smoothness parameter $\sigma = 10^{-6}$. Solving subproblem (5.6) requires computing an SVD (see (5.9)). However, we do not have to compute the whole SVD, as only the singular values that are larger than the threshold $\tau = \frac{\mu\gamma}{\max\{\gamma, \mu+\sigma\}}$ and the corresponding singular vectors are needed. We therefore use PROPACK [29], which is also used in EADM and IADM, to compute these singular values and corresponding singular vectors. To use PROPACK, one has to specify the number of leading singular values (denoted by sv_k) to be computed at iteration k. We here adopt the strategy suggested in [31] for EADM and IADM. This strategy starts with $sv_0 = 100$ and updates sv_k via:

$$sv_{k+1} = \begin{cases} svp_k + 1, & \text{if } svp_k < sv_k \\ \min\{svp_k + round(0.05d), d\}, & \text{if } svp_k = sv_k, \end{cases}$$

where $d = \min\{m, n\}$ and svp_k is the number of singular values that are larger than the threshold τ.

In all our experiments ρ was chosen equal to $1/\sqrt{m}$. We stopped ALM, EADM and IADM when the relative infeasibility was less than 10^{-7}, i.e., $\|X + Y - M\|_F < 10^{-7}\|M\|_F$.

5.3.1. Background Extraction from Surveillance Video. Extracting the almost still background from a sequence frames of video is a basic task in video surveillance. This problem is difficult due to the presence of moving foregrounds in the video. Interestingly, as shown in [6], this problem can be formulated as a RPCA problem (1.6). By stacking the columns of each frame into a long vector, we get a matrix M whose columns correspond to the sequence of frames of the video. This matrix M can be decomposed into the sum of two matrices $M := \bar{X} + \bar{Y}$. The matrix $\bar{X}$, which represents the background in the frames, should be of low rank due to the correlation between frames. The matrix $\bar{Y}$, which represents the moving objects in the foreground in the frames, should be sparse since these objects usually occupy a small portion of each frame. We apply ALM to solve (1.6) for two videos introduced in [30].

Our first example is a sequence of 200 grayscale frames of size 144×176 from a video of a hall at an airport. Thus the matrix M is in $\mathbb{R}^{25344 \times 200}$. The second example is a sequence of 320 color frames from a video taken at a campus. Since the video is colored, each frame is an image stored in the RGB format, which is a $128 \times 160 \times 3$ cube. The video is then reshaped into a 128×160 by 3×320 matrix, i.e., $M \in \mathbb{R}^{20480 \times 960}$. Some frames of the videos and the recovered backgrounds and foregrounds are shown in Figure 5.3.1. We only show the frames produced by ALM, because EADM and IADM produce visually
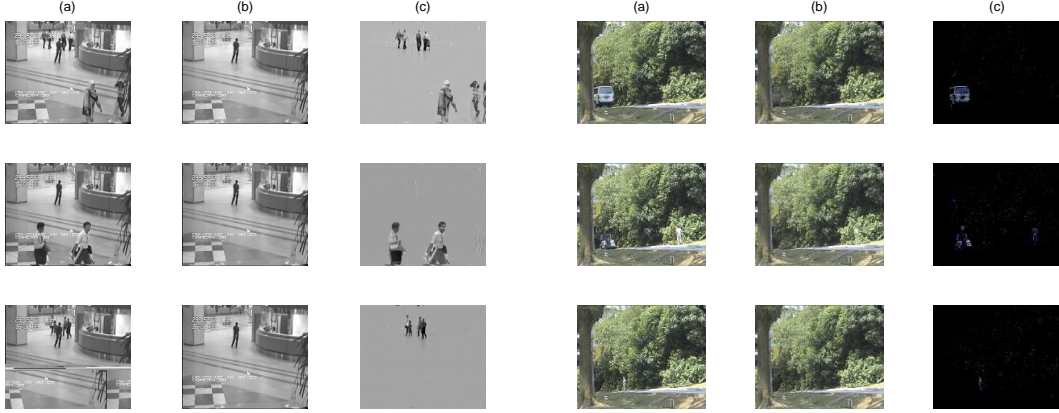
23

Fig. 5.1. *In the first 3 columns: (a) Video sequence. (b) Static background recovered by our ALM. Note that the man who kept still in the 200 frames stays as in the background. (c) Moving foreground recovered by our ALM. In the last 3 columns: (a) Video sequence. (b) Static background recovered by our ALM. (c) Moving foreground recovered by our ALM.*

identical results. From these figures we can see that ALM can effectively separate the nearly still background from the moving foreground. Table 5.1 summarizes the numerical results on these problems. The CPU times are reported in the form of $hh : mm : ss$. From Table 5.1 we see that although ALM is slightly worse than IADM, it is much faster than EADM in terms of both the number of SVDs and CPU times. We note that the numerical results in [6] show that the model (1.6) produces much better results than other competing models for background extraction in surveillance video.

TABLE 5.1
Comparison of ALM and EADM on surveillance video problems

			Exact ADM		Inexact ADM		ALM	
Problem	m	n	SVDs	CPU	SVDs	CPU	SVDs	CPU
Hall (gray)	25344	200	550	40:15	38	03:47	43	04:03
Campus (color)	20480	960	651	13:54:38	40	43:35	46	46:49

5.3.2. Random Matrix Completion Problems with Grossly Corrupted Data. For the matrix completion problem (5.11), we set $M := A + E$, where the rank r matrix $A \in \mathbb{R}^{n \times n}$ was created as the product $A_L A_R^\top$, of random matrices $A_L \in \mathbb{R}^{n \times r}$ and $A_R \in \mathbb{R}^{n \times r}$ with i.i.d. Gaussian entries $\mathcal{N}(0, 1)$ and the sparse matrix E was generated by choosing its support uniformly at random and its nonzero entries uniformly i.i.d. in the interval $[-500, 500]$. In Table 5.2, $rr := \text{rank}(A)/n$, $spr := \|E\|_0/n^2$, the relative errors $relX := \|X - A\|_F/\|A\|_F$ and $relY := \|Y - E\|_F/\|E\|_F$, and the sampling ratio of Ω, $SR = m/n^2$. The m indices in Ω were generated uniformly at random. We set $\rho = 1/\sqrt{n}$ and stopped ALM when the relative infeasibility $\|X + Y - \mathcal{P}_\Omega(M)\|_F/\|\mathcal{P}_\Omega(M)\|_F < 10^{-5}$ and for our continuation strategy, we set $\mu_0 = \|\mathcal{P}_\Omega(M)\|_F/1.25$. The test results obtained using ALM to solve (5.12) with the nonsmooth functions replaced by their smoothed approximations are given in Table 5.2. From Table 5.2 we see that ALM recovered the test matrices from a limited number of observations. Note that a fairly high number of samples was needed to obtain small relative errors due to the presence of noise. The number of iterations needed was almost constant (around 36), no matter the size of the problems. The CPU times (in seconds) needed are also reported.

TABLE 5.2
Numerical results for noisy matrix completion problems

rr	spr	iter	relX	relY	cpu	iter	relX	relY	cpu
		SR = 90%, n = 500				SR = 80%, n = 500			
0.05	0.05	36	4.60e-5	4.25e-6	137	36	3.24e-5	4.31e-6	153
0.05	0.1	36	4.68e-5	5.29e-6	156	36	4.40e-5	4.91e-6	161
0.1	0.05	36	4.04e-5	3.74e-6	128	36	1.28e-3	1.33e-4	129
0.1	0.1	36	6.00e-4	4.50e-5	129	35	1.06e-2	7.59e-4	124
		SR = 90%, n = 1000				SR = 80%, n = 1000			
0.05	0.05	37	3.10e-5	3.96e-6	1089	37	2.27e-5	4.14e-6	1191
0.05	0.1	37	3.20e-5	4.93e-6	1213	37	3.00e-5	4.66e-6	1271
0.1	0.05	37	2.68e-5	3.34e-6	982	37	1.75e-4	2.49e-5	994
0.1	0.1	37	3.64e-5	4.51e-6	1004	36	4.62e-3	4.63e-4	965

5.4. Sparse Inverse Covariance Selection. In [43] ALM method was successfully applied to the Sparse Inverse Covariance Selection problem:

$$(5.16) \qquad \min_{X \in S_{++}^n} \quad F(X) \equiv f(X) + g(X),$$

where $f(X) = -\log \det(X) + \langle S, X \rangle$ and $g(X) = \rho\|X\|_1$.

Note that in our case $f(X)$ does not have Lipschitz continuous gradient in general. Moreover, $f(X)$ is only defined for positive definite matrices while $g(X)$ is defined everywhere. These properties of the objective function make the SICS problem especially challenging for optimization methods. Nevertheless, we can still apply Algorithm 4 and obtain the complexity bound in Theorem 2.3 as follows. As proved in [33], the optimal solution X^* of (5.16) satisfies $X \succeq \alpha I$, where $\alpha = \frac{1}{\|S\|+n\rho}$, (see Proposition 3.1 in [33]). Therefore, the SICS problem (5.16) can be formulated as:

$$(5.17) \qquad \min_{X,Y}\{f(X) + g(Y) : X - Y = 0, X \in \mathcal{C}, Y \in \mathcal{C}\},$$

where $\mathcal{C} := \{X \in S^n : X \succeq \frac{\alpha}{2}I\}$. We can apply Algorithm 4 and Theorem 2.3 as per Remark 2.7. The difficulty arises, however, when performing minimization in Y (Step 5 of Algorithm 4) with the constraint $Y \in \mathcal{C}$. Without this constraint, the minimization is obtained by a matrix shrinkage operation. However, the problem becomes harder to solve with this additional constraint. Minimization in X (Step 3 of Algorithm 4) with or without the constraint $X \in \mathcal{C}$ is accomplished by performing an SVD of the current iterate Y^k. Hence the constraint can be easily imposed. Also note that once the SVD is computed both $\nabla f(X^{k+1})$ and $\nabla f(Y^k)$ are readily available (see [43] for details). This implies that either skipping or nonskipping iterations of Algorithm 4 can be performed at the same cost as one ISTA iteration.

Instead of imposing constraint $Y \in \mathcal{C}$ in Step 5 of Algorithm 4 we can obtain feasible solutions by a line search on μ. We know that the constraint $X \succeq \frac{\alpha}{2}I$ is not tight at the solution. Hence if we start the algorithm with $X \succeq \alpha I$ and restrict the step size μ to be sufficiently small then the iterates of the method will remain in $\mathcal{C}$. Similarly, one can apply ISTA with small steps to remain in $\mathcal{C}$. Note however, that the bound on the Lipschitz constant of the gradient of $f(X)$ is $1/\alpha^2$ and hence can be very large. It is not practical to restrict μ in the algorithm to be smaller than α^2, since μ determines the step size at each iteration. The advantage of ALM methods over ISTA in this case is that as soon as the $Y \in \mathcal{C}$ is relaxed ISTA can no longer be applied, while ALM/SADAL can be applied and indeed works very well. The theory in this case only

applies once certain proximity to the optimal solution has been reached. But as shown in [43], the SADAL method is computationally superior to other state-of-the-art methods for SICS.

We have also applied the FALM method to the SICS problem, but we have not observed any advantage over ALM for this particular application.

6. Conclusion. In this paper, we proposed both basic and accelerated versions of alternating linearization methods for minimizing the sum of two convex functions. Our basic methods require at most $O(1/\epsilon)$ iterations to obtain an ϵ-optimal solution, while our accelerated methods require at most $O(1/\sqrt{\epsilon})$ iterations with only a small additional amount of computational effort at each iteration. Numerical results on image deblurring, background extraction from surveillance video and matrix completion with grossly corrupted data are reported. These results demonstrate the efficiency and the practical potential of our algorithms.

References.

[1] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, *An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems*, Accepted in IEEE Transactions on Image Processing, (2009).

[2] ———, *Fast image recovery using variable splitting and constrained optimization*, preprint available at http://arxiv.org/abs/0910.4887, (2009).

[3] O. Banerjee, L. El Ghaoui, and A. d'Aspremont, *Model selection through sparse maximum likelihood estimation for multivariate gaussian for binary data*, Journal of Machine Learning Research, 9 (2008), pp. 485–516.

[4] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sciences, 2 (2009), pp. 183–202.

[5] D. P. Bertsekas, *Nonlinear Programming, 2nd Ed*, Athena Scientific, Belmont, Massachusetts, 1999.

[6] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, submitted, (2009).

[7] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, 9 (2009), pp. 717–772.

[8] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory, 52 (2006), pp. 489–509.

[9] E. J. Candès and T. Tao, *The power of convex relaxation: near-optimal matrix completion*, IEEE Trans. Inform. Theory, 56 (2009), pp. 2053–2080.

[10] P. L. Combettes, *Solving monotone inclusions via compositions of nonexpansive averaged operators*, Optimization, 53 (2004), pp. 475–504.

[11] P. L. Combettes and Jean-Christophe Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 564–574.

[12] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, SIAM Journal on Multiscale Modeling and Simulation, 4 (2005), pp. 1168–1200.

[13] D. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306.

[14] J. Douglas and H. H. Rachford, *On the numerical solution of the heat conduction problem in 2 and 3 space variables*, Transactions of the American Mathematical Society, 82 (1956), pp. 421–439.

[15] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.

[16] J. Eckstein and B. F. Svaiter, *A family of projective splitting methods for sum of two maximal monotone operators*, Math. Program. Ser. B, 111 (2008), pp. 173–199.

[17] M. Figueiredo and R. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Transactions on Image Processing, 12 (2003), pp. 906–916.

[18] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics, (2007).

[19] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite-element approximations*, Comp. Math. Appl., 2 (1976), pp. 17–40.

[20] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, SIAM, Philadelphia, Pennsylvania, 1989.

[21] D. Goldfarb and S. Ma, *Fast multiple splitting algorithms for convex optimization*, tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4570, 2009.

[22] T. Goldstein and S. Osher, *The split Bregman algorithm for L1 regularized problems*, UCLA CAM Report 08-29, (2008).

[23] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM Journal on Optimization, 19 (2008), pp. 1107–1130.

[24] B. S. He, L.-Z. Liao, D. Han, and H. Yang, *A new inexact alternating direction method for monotone variational inequalities*, Math. Program., 92 (2002), pp. 103–118.

[25] B. S. He, M. Tao, M. Xu, and X. Yuan, *Alternating direction based contraction method for generally separable linearly constrained convex programming problems*, Preprint, (2009).

[26] B. S. He, H. Yang, and S. L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, Journal of optimization theory and applications, 106 (2000), pp. 337–356.

[27] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix completion from a few entries*, IEEE Trans. on Info. Theory, 56 (2010), pp. 2980–2998.

[28] K. C. Kiwiel, C. H. Rosa, and A. Ruszczynski, *Proximal decomposition via alternating linearization*, SIAM J. Optimization, 9 (1999), pp. 668–689.

[29] R. M. Larsen, *PROPACK - software for large and sparse SVD calculations*, Available from http://sun.stanford.edu/~rmunk/PROPACK.

[30] L. Li, W. Huang, I. Gu, and Q. Tian, *Statistical modeling of complex backgrounds for foreground object detection*, IEEE Trans. on Image Processing, 13 (2004), pp. 1459–1472.

[31] Z. Lin, M. Chen, L. Wu, and Y. Ma, *The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices*, preprint, (2009).

[32] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[33] Z. Lu, *Smooth optimization approach for sparse covariance selection*, SIAM J. Optim., 19 (2009), pp. 1807–1827.

[34] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, To appear in Mathematical Programming Series A, (2009). (published online: 23 September

2009).

[35] J. Malick, J. Povh, F. Rendl, and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM Journal on Optimization, 20 (2009), pp. 336–356.

[36] R. D. C. Monteiro and B. F. Svaiter, *Iteration-complexity of block-decomposition algorithms and the alternating minimization augmented Lagrangian method*, Preprint, (2010).

[37] Y. E. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence* $\mathcal{O}(1/k^2)$, Dokl. Akad. Nauk SSSR, 269 (1983), pp. 543–547.

[38] ———, *Introductory lectures on convex optimization*, 87 (2004), pp. xviii+236. A basic course.

[39] ———, *Smooth minimization for non-smooth functions*, Math. Program. Ser. A, 103 (2005), pp. 127–152.

[40] ———, *Gradient methods for minimizing composite objective function*, CORE Discussion Paper 2007/76, (2007).

[41] D. H. Peaceman and H. H. Rachford, *The numerical solution of parabolic elliptic differential equations*, SIAM Journal on Applied Mathematics, 3 (1955), pp. 28–41.

[42] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization*, To appear in SIAM Review, (2007).

[43] K. Scheinberg, S. Ma, and D. Goldfarb, *Sparse inverse covariance selection via alternating linearization methods*, in Proceedings of the Neural Information Processing Systems (NIPS), 2010.

[44] J. E. Spingarn, *Partial inverse of a monotone operator*, Appl. Math. Optim., 10 (1983), pp. 247–265.

[45] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, preprint, National University of Singapore, (2009).

[46] P. Tseng, *Further applications of a splitting algorithm to decomposition in variational inequalities and convex programming*, Mathematical Programming, 48 (1990), pp. 249–263.

[47] ———, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities*, SIAM J. Control and Optimization, 29 (1991), pp. 119–138.

[48] ———, *On accelerated proximal gradient methods for convex-concave optimization*, submitted to SIAM J. Optim., (2008).

[49] M. Wainwright, P. Ravikumar, and J. Lafferty, *High-dimensional graphical model selection using ℓ_1-regularized logistic regression*, NIPS, 19 (2007), pp. 1465–1472.

[50] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, tech. report, Columbia University, 2009.

[51] J. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1 problems in compressive sensing*, preprint, (2009).

[52] M. Yuan and Y. Lin, *Model selection and estimation in the Gaussian graphical model*, Biometrika, 94 (2007), pp. 19–35.

[53] X. Yuan, *Alternating direction methods for sparse covariance selection*, (2009). Preprint available at http://www.optimization-online.org/DB_HTML/2009/09/2390.html.

[54] X. Yuan and J. Yang, *Sparse and low rank matrix decomposition via alternating direction methods*, preprint, (2009).